How to Generate Dynamical and Flexible Codes in Clinical Trial

Xianming (Steve) Zheng, Eli Lilly and Company

ABSTRACT

In clinical trial, we typically develop and move TFLs to production before data lock to ensure blinded condition. Consequently, there are many unknown information associated with this process, and we try our best to minimize occasions to recheck programs in production. Therefore, it requires many technical skills to achieve this goal, especially advanced macro skills. It is essential to read existing data as parameters, use those parameters as condition and pass to future codes. I will use two biggest trials in our company as example to demonstrate how to utilize those skills in real studies.

INTRODUCTION

Pharmaceutical industry is well regulated by agency, so we have very high standard in term of development of protocol and SAP, and data collection, analysis and interpretation. Among many others, the following processes ensuring integrity of studies: signing SAP before FPV, randomizing patients, blinding people involving data, moving TFLs programs to production before datalock, locking data prior to un-blind. Those good practices, however, arises many challenges for statistical analysis due to many unknown information, e.g. what variables included in model, how many dummy variables created for one class, etc.. Macro is very useful tool to handle those situations. You can read unknown information as macro and pass macro parameters to later codes. I will use three examples developed in two biggest trials in our company to demonstrate those skills.

EXAMPLE 1: STRATIFIED BY REGION

In our SAP: "the treatment-region interaction will be tested based on a likelihood ratio test for ***regions with at least 5 events***." There are six regions – Western Europe, Eastern Europe, Latin America, North America, Asia Pacific, and Africa, but two regions – Asia Pacific and Africa <5 events. So the normal approach is excluding two regions with less 5 events, creating three dummy variables and three interaction variables with treatment. The problem is due to unknown data before datalock, so the pending issues such as: how many regions, which region in the model; how many dummy variables to create and how to create; how to put dummy variables and interactions in the model. The solutions are following:

- PROC FREQ → IF COUNT >= 5 → merge back with original data to satisfy condition.
- Read how many total regions and which one:

```
CALL SYMPUT('REG'||LEFT(_N_), REGION);
CALL SYMPUT('OBS', _N_);
```

- Creating dummy variables and interactions:

```
%LET DUMMYREG=;   %LET DUMMYINT=;   %LET M=1;

%DO %UNTIL(%EVAL(&M)=%EVAL(&OBS));
   DATA BCREG;
       SET BCREG0;
       BY REGION;
       RETAIN COUNT 0;
       IF FIRST.REGION THEN COUNT=COUNT+1;
       IF COUNT=&M THEN D&M=1;  ELSE D&M=0;
       INT&M=D&M * TRT;
    RUN;
     %LET DUMMYREG=&DUMMYREG    D&M;
     %LET DUMMYINT=&DUMMYINT    INT&M;
     %LET M=%EVAL(&M+1);
%END;
```

- fit into the model:

```
 PROC PHREG DATA=BCREG;
     MODEL YEAR*CENSOR(0)=TRT &DUMMYREG &DUMMYINT    /TIES=EXACT;
RUN;
```

- Model II: test treatment effect in each region:

```
%LET X=1;
 %DO %UNTIL(%EVAL(&X)=%EVAL(&OBS+1));
     PROC PHREG DATA=BCREG(WHERE=(REGION="&&REG&X"));
         MODEL YEAR*CENSOR(0)=TRT /TIES=EXACT RL;
     RUN;

    %LET X=%EVAL(&X+1);
 %END;
```

EXAMPLE 2. ADJUSTED FOR RISK FACTORS

In this example, we want to evaluate treatment effects on invasive breast cancer after adjusting baseline risk factors such as age, gail score, concomitant medication, etc.. The normal procedure will be 1) univariate Cox model for each risk factor; 2)multivariate model with significant factor using stepwise selection process: 3)final model. The problem is 1) result driven procedure; 2)In three group factors, one dummy variable significant and the other not, should included both dummy variable; 3)how many factors in the model, which one, how to put in footnote? Here are ways we did:

- In univariate model, pick p-Value < 0.05 (if factors with two dummy var., then pick smaller p-Value, so it will include both dummy var.).
- Generated macro from dataset:

```
data _null_;
   set all nobs=count;
   call symput('rf'||left(_n_), variable);
   call symput('obs',count);
 run;
```

- In Multivariate Cox model:

```
proc phreg data=bc ;
   model YEAR*CENSOR(0)=
        %do num=1 %to &obs;
              &&rf&num
        %end;
    / ties=exact selection=s slstay=0.05 slentry=0.10 ;
run;
```

- Risk Factors in Footnote:

1) not just variable name
2) variable label. This is great way in case of many variables.

```
   call symput ("varlabel",VLABEL(&X));
```

3) generated text:

```
 &if "&&rf&num"="age" %then   %let fnotevar=&fnotevar  %str(@2  " Age (<=65, >65)");
```

EXAMPLE 3. AUTO RENAME LIBRARY VARIABLES

In Prasugrel regulatory response, FDA required: no underscore(_) in variable name, variable <= $8, etc.. Team did what FDA required, but wanted to make sure two versions were same except the known changes. We took proactive procedures, which make known changes so PROC COMPARE finds nothing difference, otherwise there will be big documents to review considering huge datasets.

- Catch variable names from CONTENTS and make required changes:

```
 PROC CONTENTS data=xx out=xv(keep=name);
 DATA XV1;  SET XV;
     VNAME=COMPRESS(NAME,'_');
     IF LENGTH(TRIM(LEFT(VNAME)))<=8 THEN  TVNAME=VNAME;
    TVNAME=PUT(VNAME,$8.);
 RUN;
```

- Generate macro for old and new variables:

```
data _null_;
   set av nobs=count;
   call symput('ovar'||left(put(_n_,4.)), name);
   call symput('nvar'||left(put(_n_,4.)), tvname);
   call symput('obs', put(count, 4.));
```

run;

- Rename dataset variables:

```
%macro varname;
   %do num=1 %to &obs;
            &&ovar&num=&&nvar&num
        %end;
%mend varname;
%macro varrename;
         data x; set a (rename=(%varname));     run;
 %mend;

 %varrename;
```

CONCLUSION

In this tough environment, it is always good practice to avoid hardcode or recheck production programs. There will be many challenges ahead, especially for relative new programmers. Macro, particularly macro parameter and loop, is good tool to generate dynamical and flexible codes. Those three examples will be very helpful to understand advanced macro skills and how to implement in real study.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Xianming(Steve) Zheng
Eli Lilly and Company
450 Madison Ave
Indianapolis, IN 46225
Work Phone: 317-433-9044
Fax: 317-276-6244
E-mail: Zhengxi@lilly.com